# Clickbait Detection Using Machine Learning

Yuya Ong, Vince Trost, Sanchit Goel, Wentao Yan
DS 310: Machine Learning

## Abstract

Clickbait in the mainstream media has become an ubiquitous centerpiece for attracting more views and hits from users to enable them to click on the article for increased traffic and revenue gain. In our project, we have experimented with various different features and combinations of different classification models which aims to predict whether a given article is considered clickbait or non-clickbait.

## Introduction

Clickbait media, in recent years, have garnered attention from various media sites and social media networks as they have become a pervasive part of the internet. Its primary objective is to simply get users to open the article link, which allows site owners and content curators to gain attraction of new users to enable them with greater visibility and income from generated ad revenues presented to the users. Often times, the titles that are displayed to the users are crafted in such ways that purposefully attract attention. However, the article content which accompanies the title often does not reflect or sometimes contradict the content which is promised by the titles - therefore "baiting" the users into clicking the article. This has become a significant problem with the proliferation of other major issues such as fake news, where sensationalization of content has become a primary objective for authors to unethically manipulate the digital information landscape and potentially drive misinformation within the online communities.

In our work, we have explored various techniques and methodologies to combat against clickbait sources through building a binary classification model which detects whether the given content is clickbait or non-clickbait. We explored various different types of features including both hand-crafted grammar based heuristics, frequency and count-based heuristics, to word embeddings to model the semantics of the given textual features. Our focus in this study was based primarily on studying the effect of how certain features affect whether a given article is considered to be clickbait or non-clickbait, as well as testing our hypothesis with respect to the features we have constructed and how they have performed with the models we built using the recommended features from our statistical analysis. Our contributions in this project are as follows:

- Statistical analysis of 119 handcrafted features based on relevance, correlation, and importance through various methods
- Implementation of various word vectorization techniques including Bag-of-Words, TF-IDF, and Word Embeddings
- Various model implementations using the above listed features, with two models achieving high metrics.
- Present a theoretical modeling framework for training a topic-invariant model for detecting clickbait articles.

All preliminary analysis and feature engineering notebooks, implementations for every model, individual model metrics, as well as preprocessing scripts are provided in our Github repository for reference and reproducibility of our experiments.

**Link**: https://github.com/yutarochan/clickbait

## Dataset

### Dataset Description

The dataset provided to us were based on a scraped Twitter feed data which only contains articles people have shared. In the feature set, they included the corresponding article headlines, the content of those articles, and associated media, metadata, keywords associated with the articles, corresponding post text, and the timestamp. These were labeled by the Amazon Mechanical Turk workers where five annotations were given to an article as to whether or not it was clickbait, and the majority vote decided its classification. There were over 17,581 articles in the training set and 4,416 articles in the test set. Within the test set, there were a total of 13,148 non-clickbait articles while there were 4,433 clickbait articles in total - which indicates that for every 3 non-clickbait article there is 1 clickbait article in this dataset. For each article, the provided dataset also included various labels, including but not limited to, the truth class, which was defined to be our objective label for the model to learn, as well as each of the five annotators individual annotation values, mean, median, and mode.

**Dataset Analysis**

Prior to performing modeling, we have conducted various statistical properties of our dataset to obtain a better sense of the overall distribution of the various features and attributes provided to us. Using this information has helped to both model and provide recommendations in ways to improve our modeling methods. In each of our analysis we evaluate the statistical parameters for the overall distribution of the dataset, as well as the individual parameters for the clickbait and the non-clickbait to empirically see if there exists any significant differences in the parameters.

First, we evaluated the general statistics of the labels within our dataset and found that across all the metadata such as the mean, variance, and the kew to be fairly consistent where the mean falls within the lower 0.27 to 0.33 ranges, indicating that much of the articles were slightly confidently rated non-clickbait and the variance to be fairly within a tighter range between 0.06 to 0.14. Thus, given this information, we should expect that a strong performing model should follow similar heuristics where the majority of the classifications would adhere to higher confidence with respect to true negative values.

Next, we analyzed the distributions of the article's headline for which the majority of the parameters were consistent across all levels - however, we found that the variance of the headlines use of casing (upper vs lower) differed significantly between both the clickbait and non-clickbait headlines indicating the potential difference in the casing style of the text.

Then, we performed an analysis of the frequency of the terms found in the headlines, post, and keyword distributions. A key observation that we have found in this particular analysis is that the words that were highly frequent were mostly topics which are based on sensationalized keywords that were most trending during the duration of the data mining process of the given dataset, such as: "Trump", "Police", "Woman", "China", etc. These are topics often brought up in major media sources which can or cannot be considered clickbait, as often times many major social media sources and media journalists often curate content around current trending areas which often does have a higher probability of containing clickbait material more so than a niche area topic. Although this feature may be of a potential critical indicator which may be highly relevant in the modeling process, we suspect several different issues which will be further discussed in the *Discussion and Analysis* section of our paper.

## Feature Extraction

In this this work, we have constructed various features including both handcrafted features as well as various word vectorization techniques. Prior to specific feature extraction methods, we employed various preprocessing of the data across each of the data. For each headlines, we perform standard tokenization, normalization (i.e. lower case) and removal of punctuation. For each body text and post text information, we also perform a similar operation as the former, however, we have not employed any other methods such as stopword removal nor stemming as we believed that the usage style of these words reflect whether the article's content is considered clickbait or non-clickbait. In this section we will briefly cover some of the methods we have utilized to construct those features.

**Handcrafted Features**

In our work, we have hand constructed 119 features which are all listed in **Table 1,** for which each of the numbers in the features correspond to the index of the feature vector (and not necessarily ranked by feature importance) we have used to train our model with. Note that our table uses a 0 based indexing of the features and below the table includes the type of text these features were extracted from indicated by special characters next to each category. Most features included attempts of reproducing NLP based heuristics from existing work, including top 60 features proposed by Cao et. al [1] as well as other features including various n-gram, POS ratio to the selected paragraph text token counts, as well as various readability metrics on the selected paragraphs found in various reading comprehension studies within the area of linguistics.

**Table 1:** 119 Handcrafted Feature Sets

| | | | | | |
|---|---|---|---|---|---|
| 0 | Number of NNP ♠ | 41* | Ratio of RBS to Text ♦ | 82* | Number of PRP ♠ |
| 1 | Automated Readability Index (ARI) ♦ | 42* | Ratio of RP to Text ♦ | 83 | Number of PRP ♠ |
| 2 | Avg. Letter Per Word ♦ | 43* | Ratio of SYM to Text ♦ | 84* | Number of VBZ ♠ |
| 3 | Avg. Sentence Length ♦ | 44 | Ratio of TO to Text ♦ | 85 | POS 3-Gram NNP NNP VBZ ♠ |
| 4 | Avg. Sentence Per Word ♦ | 45* | Ratio of UH to Text ♦ | 86 | POS 2-Gram NN IN ♠ |
| 5 | Avg. Syllables Per Word ♦ | 46 | Ratio of VB to Text ♦ | 87* | POS 3-Gram NN IN NNP ♠ |
| 6 | Character Count ♦ | 47 | Ratio of VBD to Text ♦ | 88* | POS 2-Gram NNP . ♠ |
| 7* | Coleman-Liau Index ♦ | 48* | Ratio of VBG to Text ♦ | 89* | POS-2-Gram PRP VBP ♠ |
| 8 | Dale-Chall Readability Score ♦ | 49* | Ratio of VBN to Text ♦ | 90* | Number of WP ♠ |
| 9 | Number of Difficult Words ♦ | 50* | Ratio of VBP to Text ♦ | 91* | Number of DT ♠ |
| 10 | Flesch-Kincaid Grade ♦ | 51* | Ratio of VBZ to Text ♦ | 92 | POS 2-Gram NNP IN ♠ |
| 11 | Flesch Reading Ease Score ♦ | 52* | Ratio of WDT to Text ♦ | 93 | POS 2-Gram IN NNP NNP ♠ |
| 12 | Gunning Fog Score ♦ | 53* | Ratio of WP to Text ♦ | 94* | Count POS Pattern POS ♠ |
| 13 | Lexicon Count ♦ | 54* | Ratio of WP$ to Text ♦ | 95* | POS 2-Gram IN NN ♠ |
| 14 | Linsear Write Formula ♦ | 55* | Ratio of WRB to Text ♦ | 96* | Keywords to Post Text Match |
| 15 | Polysyllabic Count ♦ | 56 | N/A | 97* | Count of ‘,’ ♠ |
| 16* | Sentence Count ♦ | 57 | Unigram (1-gram) ♦ | 98* | POS 2-Gram NNP NNS ♠ |
| 17* | SMOG Index ♦ | 58 | Bigram (2-gram) ♦ | 99* | POS 2-Gram IN JJ ♠ |
| 18 | Syllable Count ♦ | 59 | Trigram (3-gram) ♦ | 100* | POS 2-Gram NNP POS ♠ |
| 19 | Mean Word Length ♦ | 60 | 4-gram ♦ | 101* | Number of WDT ♠ |
| 20 | Ratio of CC to Text ♦ | 61 | 5-gram ♦ | 102 | POS 2-Gram NN NN ♠ |
| 21* | Ratio of CD to Text ♦ | 62 | Maximum Word Length ♥ | 103* | POS 2-Gram NN NNP ♠ |
| 22 | Ratio of DT to Text ♦ | 63* | Stop Words Ratio ♥ | 104* | POS 2-Gram NNP VBD ♠ |
| 23* | Ratio of EX to Text ♦ | 64 | N/A | 105 | Number of RB ♠ |
| 24* | Ratio of FW to Text ♦ | 65 | Unigram (1-gram) ♥ | 106 | POS 3-Gram NNP NNP NNP ♠ |
| 25 | Ratio of IN to Text ♦ | 66 | Bigram (2-gram) ♥ | 107* | POS 3-Gram NNP NNP NN ♠ |
| 26 | Ratio of JJ to Text ♦ | 67 | Trigram (3-gram) ♥ | 108 | Number of RB ♠ |
| 27* | Ratio of JJR to Text ♦ | 68 | 4-gram ♥ | 109 | Number of VBN ♠ |
| 28* | Ratio of JJS to Text ♦ | 69 | 5-gram ♥ | 110* | POS 2-Gram VBN IN ♠ |
| 29* | Ratio of LS to Text ♦ | 70 | Number of Tokens ♠ | 111 | POS 2-Gram JJ NNP ♠ |
| 30* | Ratio of MD to Text ♦ | 71 | Number of Words ♠ | 112 | Number of RBS ♠ |
| 31 | Ratio of NN to Text ♦ | 72* | POS 2-Gram NNP NNP ♠ | 113 | Number of VBN ♠ |
| 32 | Ratio of NNS to Text ♦ | 73* | Whether Starts with Number ♠ | 114 | POS 2-Gram VBN IN ♠ |
| 33* | Ratio of NNP to Text ♦ | 74 | Number of IN ♠ | 115 | POS 2-Gram JJ NNP ♠ |
| 34* | Ratio of NNPS to Text ♦ | 75 | POS 2-Gram NNP VBZ ♠ | 116 | POS 3-Gram NNP NN NN ♠ |
| 35* | Ratio of PDT to Text ♦ | 76* | POS 2-Gram IN NNP ♠ | 117 | POS 2-Gram DT NN ♠ |
| 36* | Ratio of POS to Text ♦ | 77* | Number of WRB ♠ | 118 | Existence of EX ♠ |
| 37 | Ratio of PRP to Text ♦ | 78 | Number of NNP ♠ | | |
| 38* | Ratio of PRP$ to Text ♦ | 79* | Title Starts with 5W1H ♠ | | |
| 39 | Ratio of RB to Text ♦ | 80* | Whether Exists QM ♠ | | |
| 40* | Ratio of RBR to Text ♦ | 81* | Number of This/These NN ♠ | | |

**Feature Index:**

♠ = Target Title ♦ = Target Paragraphs ♥ = Post Text * = VIF Proposed Features

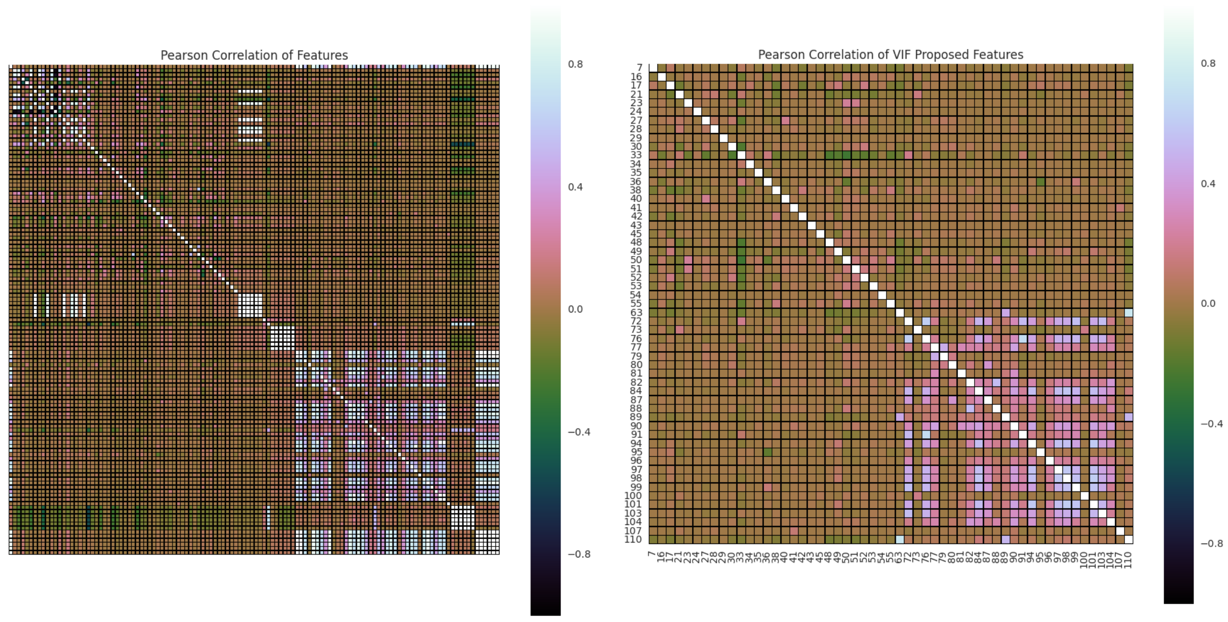
**Analysis of Handcrafted Features**

Given the proposed feature sets we have extracted from the given dataset, we were specifically interested in analyzing the relative relevancy of each of the handcrafted features with respect to the corresponding annotations for each of the samples. We now describe the various methodologies we employed to evaluate the significance and importance each feature has to determine the predictive power of the feature which contributes to the overall model. Although due to time constraints we have not been able to implement individual models based on these features, but we have identified the key features which we would have used.

Variance Inflation Factor

One of the major methods we have utilized to reduce the dimensionality space of our proposed feature vector was through identifying features which exhibit a highly multicollinear behavior between other features, which increases the standard errors of the corresponding parameters for a feature. To combat against this, we performed a systematic methodology to evaluate the Variance Inflation Factor, which provides an index for evaluating the severity of multicollinearity using the ordinary least squares method. We iteratively drop the corresponding features based on a predefined threshold as a hyperparameter and eliminate those variables from the feature set. For our analysis, we utilized a threshold to eliminate any VIF values which exceeds 5.0. As a result of our process, we were able to subsequently reduce our feature space down to 55 different features from our initial 120. Remaining features from this process are indicated as a '*' next to the corresponding indices for each of the respective features in **Table 1.** In **Figure 1,** we have plotted a heatmap of the Pairwise Pearson Correlation between the feature sets for first all of the proposed 120 features (left), and the feature sets proposed by our systematic VIF based process (right).

**Figure 1:** Pairwise Pearson Correlation



Ranked Feature Selection

We performed the following approaches, using various different modeling methodologies to perform an aggregate analysis of the features and produce an average ranking based on these metrics to determine the most important features. The result rankings of the top 15 features are listed on **Table 2**, which is ranked by the mean.

- Stability Selection using Randomized Lasso Method by Meinshausen et. al [4]
- Recursive Feature Elimination (RFE)
- Linear Regression Based Feature Ranking
- Lasso Regression Based Feature Ranking
- Ridge Regression Based Feature Ranking
- Random Forest Based Feature Ranking

**Table 2:** Top 15 Features Based on Various Correlation Metrics - Ranked by Mean

| Feature | Lasso | Linear Reg | Rand Forest | RFE | Ridge | Stability | Mean |
|---|---|---|---|---|---|---|---|
| 47 | 0.0 | 0.0 | 0.91 | 0.8 | 1.0 | 0.0 | 0.45 |
| 25 | 0.0 | 0.0 | 0.95 | 0.75 | 0.6 | 0.0 | 0.38 |
| 26 | 0.0 | 0.0 | 1.0 | 0.54 | 0.76 | 0.0 | 0.38 |
| 38 | 0.0 | 0.0 | 0.95 | 0.76 | 0.6 | 0.0 | 0.38 |
| 49 | 0.0 | 0.0 | 0.97 | 0.83 | 0.51 | 0.0 | 0.38 |
| 32 | 0.0 | 0.0 | 0.98 | 0.5 | 0.77 | 0.0 | 0.37 |
| 116 | 0.0 | 1.0 | 0.16 | 0.97 | 0.03 | 0.0 | 0.36 |
| 20 | 0.0 | 0.0 | 0.98 | 0.62 | 0.44 | 0.0 | 0.34 |
| 22 | 0.0 | 0.0 | 0.89 | 0.71 | 0.37 | 0.0 | 0.33 |
| 39 | 0.0 | 0.0 | 0.96 | 0.7 | 0.35 | 0.0 | 0.33 |
| 31 | 0.0 | 0.0 | 0.99 | 0.34 | 0.62 | 0.0 | 0.32 |
| 50 | 0.0 | 0.0 | 0.85 | 0.72 | 0.33 | 0.0 | 0.32 |
| 52 | 0.0 | 0.0 | 0.79 | 0.87 | 0.18 | 0.0 | 0.31 |
| 114 | 0.31 | 0.37 | 0.15 | 0.98 | 0.06 | 0.0 | 0.31 |
| 21 | 0.0 | 0.0 | 0.92 | 0.67 | 0.21 | 0.0 | 0.3 |

**Word Vectorization Features**

We have also employed various word embedding techniques based on Bag-of-Words, TF-Idf as well as Word Embedding techniques trained only within the corpus set of the given text data. We then have employed each of these features through various different dimension which were then subsequently fed into the model for evaluation. For each corresponding word vectorization based model, we defined the dimension of the feature vector as our hyperparameter for the model. In our experiments we attempted various different hyperparameters to evaluate their effect on the defined metrics.

For word embeddings, we employed an extension of the Word2Vec model by Mikolov et. al [5] and employed using a Document Vector based model by Le et. al [3] instead, which can generate representation of texts for various lengths. All word embedding based models we have trained are based only on the corpus of words we were trained on and did not employ and pre-trained models. During our training process for when training our embedding models, we only performed training of the word embedding models from the training sample subset defined by the cross validation and took the testing set and performed an inference operation on the raw text to obtain a vector representation of the text prior to feeding the feature vector into the model.

## Experiments

We tried multiple experiments to test the performance of many different models using different preprocessing techniques. Some of the the experiments we performed, deemed best by the scores based on their respective feature, hyperparameters and models:

- Hand-crafted Features (119) to XGBoost Logistic Regression
- Hand-crafted Features (119) to Random Forest Binary Classification
- Hand-crafted Features (119) to Naive Bayes
- Bag-of-Words (150 dim) to Random Forest
- Bag-of-Words (400 dim) to Naive Bayes
- Bag-of-Words (100 dim) to Support Vector Machine
- Baseline Hand-crafted features to Stratified K-Fold
- TF-IDF to Logistic Regression (scikit)
- Bag-of-Words (150 dim) to Logistic Regression (scikit)
- Doc2Vec Headline + Article (400 dim) to Logistic Regression (scikit)
- Doc2Vec Headline (100 dim) to Logistic Regression (scikit)
- TF-IDF Naive Bayes
- TF-IDF to XGBoost Logistic Regression
- Doc2Vec Headline + Article (350 dim) to Logistic Regression (scikit)

For each experiment, we utilize a 10-fold stratified cross-validation technique to mitigate class imbalances with shuffling of data per individual models. For each model, we evaluate them on the standard metrics including accuracy, precision, recall, F-measure, area under the curve (AUC), as well as the Fleiss-Kappa metric. Within each model, we have experimented with various hyperparameter tuning by a simple systematic grid-search based approach - mostly applicable to the word vectorization features. We employ a standard brute-force search feature where we have conducted various experiments across each different dimensions - out of which the best dimensions with the highest and most acceptable scores have been chosen to be benchmarked against other feature and classification models we have produced as shown in **Table 3**.

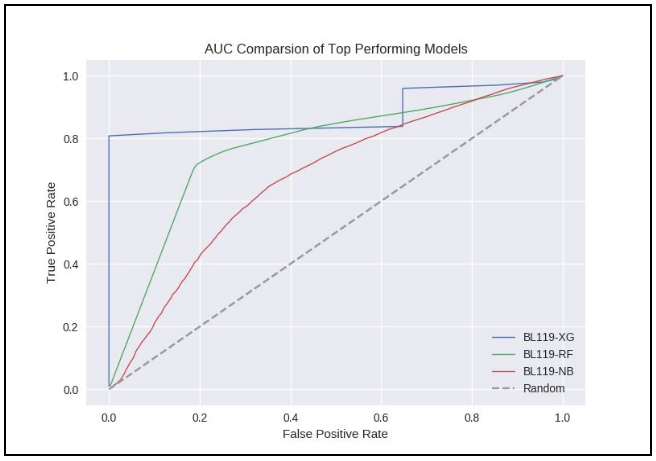
## Results

In this section we present our results from our experiments. For the purpose of conserving space, we have only selected to present the top models with the best hyperparameters, however in our full repository, we have retained all of the results of the experiments. **Table 3** shows our top ranking models, while **Figure 2** presents the AUC plot of the each corresponding each model as a comparison.

**Table 3:** Top Ranking Models Produced

| Model ID | Accuracy | Precision | Recall | F-Measure | AUC | Kappa |
|---|---|---|---|---|---|---|
| **baseline119-XG** | **0.99977149** | **0.99977728** | **0.99933135** | **0.9995538** | **0.877107261** | **0.999400239** |
| baseline119 - RF | 0.97897811 | 0.96926066 | 0.94804476 | 0.9585074 | 0.770102867 | 0.944433255 |
| baseline119 - NB | 0.74083154 | 0.47427996 | 0.15185233 | 0.2258273 | 0.673705429 | 0.121191305 |
| bow-150_randforest | 0.70837694 | 0.28951669 | 0.10799286 | 0.1569970 | 0.509432017 | 0.023748341 |
| bow-400_nb | 0.4089669 | 0.25556859 | 0.70128378 | 0.3739772 | 0.506139352 | 0.007867386 |
| bow-100_svm | 0.58056880 | 0.25729478 | 0.3514759 | 0.2963501 | 0.504782150 | 0.008451304 |
| baseline5 + SKF | 0.71378452 | 0.27842952 | 0.08255292 | 0.1244736 | 0.504579528 | 0.011956765 |
| tf-idf_logistic | 0.58472196 | 0.25625919 | 0.34056032 | 0.2922417 | 0.503818144 | 0.006710148 |
| bow-150_logistic | 0.52272298 | 0.25519235 | 0.46522399 | 0.3293479 | 0.503685863 | 0.005799053 |
| d2v-ha-400_logistic | 0.35567052 | 0.25318721 | 0.79788736 | 0.3833660 | 0.502031326 | 0.002521641 |
| d2v-100_logistic | 0.61833489 | 0.25214051 | 0.26807730 | 0.2412830 | 0.502005089 | 0.003623486 |
| tf-idf_nb | 0.49900446 | 0.25319364 | 0.50606321 | 0.3373458 | 0.501326708 | 0.002087997 |
| tf-idf_xgboost | 0.74728291 | 0.05 | 0.00045362 | 0.0008981 | 0.499770854 | -0.00068287 |
| d2v-ha-350_logistic | 0.30931361 | 0.25116653 | 0.87783065 | 0.3901459 | 0.497927487 | -0.00240277 |

**Figure 3:** AUC Plot of Top Three Performing Models



## Analysis and Discussion

Based on our experiments, we have found the best model being the 119-feature XGBoost model having very high metrics across the board - however, we suspect that XGBoost may have completely overfitted the data and therefore has the highest risk of performing poorly against new samples. However, on the other hand, we see that the Random Forest and the Naive Bayes method of the 119 hand crafted features performs significantly well against all other word vectorization methods. However, we also suspect that the Random Forest model may have potentially overfitted and should be further evaluated through analyzing the underlying structure generated by the model. To improve the overall performance of the model, it may have been best to apply a form of bagging or some other regularization factor to reduce the risk of our model from overfitting.

Apart from the handcrafted models, the majority of all the word vectorization based models which are all corpus dependent performed very poorly against all subsets of the cross validation sets with the overall AUC hovering around 0.5. However, one interesting observation to note is that the models we have produced often suffer a trade-off between accuracy and recall - which indicates a very interesting hypothesis from this behavior. We believe that by training the word vectorization model only on the given set of corpus only limits the representation that the model learned only on the set of data we provided. Since this data is based on scraped data, which only samples from a list of limited topics and therefore does not produce a model which is robust model which is topic invariant. In other words, a model which only relies only on a corpus based model are very weak in regards to detecting clickbait articles - making it more so a topic modeling problem than a clickbait problem. Therefore, even if we attempted to utilize these features within a Deep Learning based setting, we would see very similar results and would yield poor performance overall due to this specific bottleneck. However, it is not to say that a given topic is entirely non-relevant to the problem of clickbait as often the subject of clickbait is often associated with very sensationalized and highly trending topics like “Trump”, “Violence”, “Police”, etc. which explains the high variance and low accuracy behavior patterns found in our models. To potentially improve end-to-end word vectorization based models, we believe that using a pre-trained word embedding model from a large scale corpus base such as the Google News or Wikipedia dataset would be sufficient in capturing the various topic semantics necessary to balance out the accuracy and recall of the model. However, this method may still suffer from low precision due to the constantly changing semantics of topics in the space of popular media, thus consistently requiring re-training the embedding model in an online fashion - if this were to be implemented in a real-world setting.

From these experiments, we can assess that both the underlying latent grammatical structure as well as the topical semantics are very important features. However, designing such models which make use of both is most likely the next challenge that will have to be considered going forward with building a very robust model - especially when the topics are consistently changing over time.

## Future Work

In this section we describe at a very high level a Deep Learning based topic invariant modeling process for training a robust clickbait classifier based on a Generative Adversarial Network architecture proposed by Goodfellow et. al [2]. To build a model which detects clickbait articles, we believe that this model must satisfy two different properties: i) the model must be resilient and robust against any changes in the semantics of the given topic and ii) learns the underlying latent lexical and grammatical structure of clickbait material without the sacrifice of compromising the semantics. In the space of detecting clickbait based materials, we also suffer from the lack of labeled data (especially identifying clickbait than non-clickbait articles) as it is very time consuming and expensive to curate and collect such labels from a crowd-sourced approach. Furthermore, as we have seen in our dataset, the proportion of clickbait articles found are greater than the number of non-clickbait articles, therefore suffering from a significant amount of data imbalance.

Therefore, we propose a GAN based model known as **BaitGAN**, which comprises of two components: the *discriminator*, $D(x)$, or the classifier model which detects whether a given article is clickbait or non-clickbait, and two different recurrent neural network based *generative* models $G_{CB}(x)$ and $G_{NCB}(x)$, which generates clickbait and non-clickbait articles respectively. The objective of $D(x)$ is to correctly classify whether the given input, whether it came from the original distribution of articles or from the generative models, is to detect correctly whether the article is clickbait or non-clickbait. On the other hand, the objective of $G_{CB}(x)$ and $G_{NCB}(x)$ would be based on the cost function of respectively getting $D(x)$ to correctly classifying the generated samples they produced are clickbait/non-clickbait and thus reinforcing the model to learn the underlying grammatical structure of both clickbait and non-clickbait articles. The underlying training of this model would be based on training a robust classifier while also simultaneously providing a method of effectively performing oversampling methods which best emulates the underlying lexical and grammatical structure of a clickbait and non-clickbait sample while using a method to generate topic invariant clickbait samples, and making use of a hybrid approach for using unlabeled data to train the model based on both supervised and unsupervised approach for using a pre-trained classifier to increase the number of labeled sample from a sample of unlabeled data.

## Conclusion

In this report, we have presented our methodology, analysis, and recommendations based on empirical evidence and work on detecting clickbait versus non-clickbait articles. The miniscule amount of text to work with in the headlines and the imbalance of the dataset has made this problem especially challenging in the context of building a much more robust model. In our work we have produced various different models along with an in depth analysis of the different features we have proposed and their overall contribution towards the detection of clickbait materials. We have found in this process that both semantic and grammar based heuristics are critical features to keep in mind when performing modeling on clickbait detection to mitigate a good balance between accuracy, precision, recall and AUC.

## References

1. Cao, Xinyue, Le, Thai, and Zhang, Jason Jiasheng. Machine Learning Based Detection of Clickbait Posts in Social Media. arXiv preprint. arXiv:1710.01977, 2017.
2. Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In Advances in Neural Information Processing Systems, pp. 2672–2680, 2014.
3. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053, 2014.
4. Meinshausen, Nicolai and B¨uhlmann, Peter. Stability Selection. 2009.
5. Mikolov, T., Chen, K., Corrado, G., & Dean, J. Efficient estimation of word representations in vector space. In ICLR, 2013.